

Árvore

↳ Hierarquia, composição

raiz → modo de origem, subárvore → árvore dentro da árvore, usa um modo como sub-raiz
floresta → conjunto de árvores, cada uma com sua raiz
caminho → sequência de nodos. Comprimento do caminho é o nº de arestas
pai → vértice com aresta que incide nele; filho → vértice p/ onde esse vértice ^{tem arestas} incidentes
grau → nº de filhos; irmãos → filhos do mesmo pai (com mesmo nível)

Árvore Binária

↳ sempre possui no máximo 2 filhos

↳ geralmente usada p/ facilitar a pesquisa que segue um critério (ABP)

- Na ABP, pode seguir 3 tipos de busca:

1. Pré-ordem: escreve logo quando entra no nodo, depois visita os filhos

2. Em ordem: visita o filho da esquerda, escreve, visita o da direita

3. Pós-ordem: visita ambos os filhos antes de escrever

→ comum em DFS (depth first search ou busca em profundidade)

→ pode-se usar BFS (breadth first search ou busca em amplitude) também, que usa o nível como referência, pegando cada vértice

→ em ordem pela esquerda coloca na ordem crescente, pela direita na decrescente

→ pré pela esquerda é o inverso do pós pela direita (pós direita é o inverso ^{do pré pela esquerda})

- Na inserção, o novo nodo é sempre folha

- Na remoção:

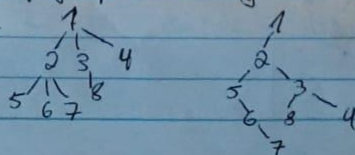
↳ folha: remove o nodo

↳ 1 filho: o filho e toda sua sub-árvore ocupam seu lugar e o nodo é removido

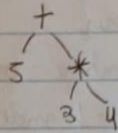
↳ 2 filhos: vai p/ o filho da esquerda e procura pelo maior filho dele (folha

4ª à direita do nodo à esquerda do removido). A folha ocupa o lugar do nodo

→ uma árvore m-ária pode ser representada por uma binária; à esquerda ficam os filhos do nodo (pega apenas o 1º filho de cada nodo) e à direita ^{do pai} ficam os irmãos. EX:



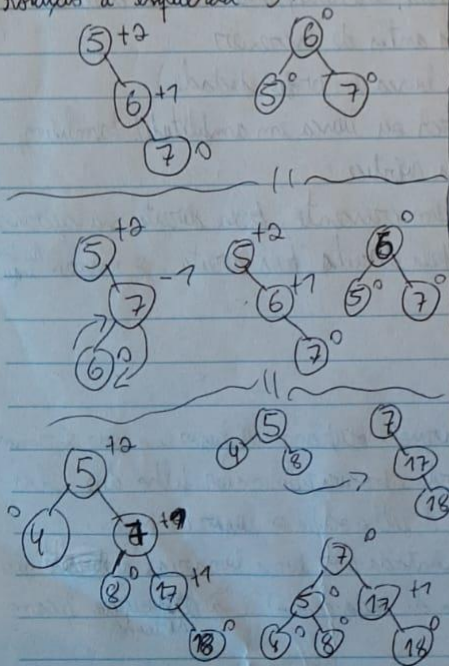
Interpretador de expressões \rightarrow raiz é operador; operadores não sempre não-afixos; operandos são folhas. A notação é escrita em pré-ordem. EX: +5*34 notação polonesa



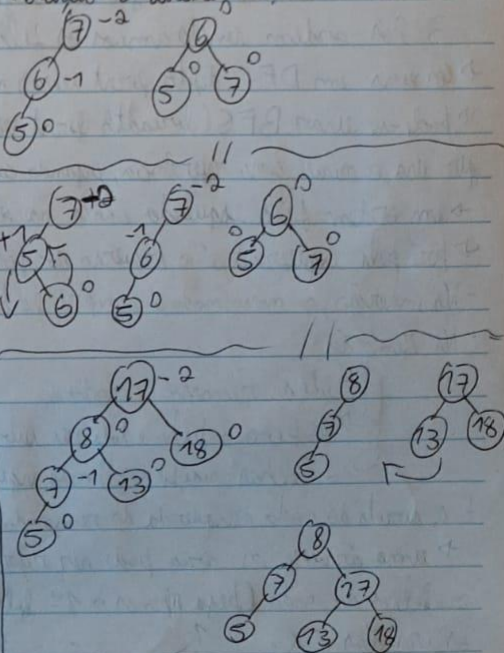
AVL \rightarrow árvore binária balanceada, quando f/ cada nó, a diferença de altura entre um subárvore esquerda e direita é no máximo 1. Essa diferença é chamada de fator de balanceamento (altura direita - altura esquerda).

- \rightarrow reduz as buscas imensamente, mas aumenta o custo nas inserções e remoções.
- \rightarrow quando um nó é adicionado e viola o balanceamento, é necessário restaurar o balanceamento através de rotações.

Rotação à esquerda \hookleftarrow :



Rotação à direita \hookrightarrow :





Universidade de Caxias do Sul
Área do Conhecimento de Ciências Exatas e Engenharias
Disciplina: FBI4005AB - Algoritmos e Estrutura de Dados I
Professora: Helena Graziottin Ribeiro
Segunda Avaliação - 03/07/2023

Nome: Eduardo Elberholt Pereira

10,0

Para as implementações, considere como estrutura de árvore binária (ou de ABP), para C e Java:

```
struct nodo{  
    int n; // ou char n  
    struct nodo *esq;  
    struct nodo *dir;  
};  
typedef struct nodo NodoA;
```

```
Class NodoA{  
    int n; // ou char n  
    NodoA esq;  
    NodoA dir;  
  
    NodoA (...){ } ...  
}
```

QUESTÃO 1) (2,0 PONTOS) Considere as regras de uma ABP (Árvore Binária de Pesquisa):

a) **Desenhe uma ABP**, cujos nodos armazenam nomes, sabendo que:

- o percurso central (à direita) corresponde a: **toco – raio – muro – mato – luta – feio – copa – boa – bela**
- o percurso pré-fixado (à esquerda) corresponde a: **copa – boa – bela – luta – feio – muro – mato – raio – toco**

b) **Desenhe a ABP** resultante da remoção de **copa** na ABP que você desenhou no item 1 a).

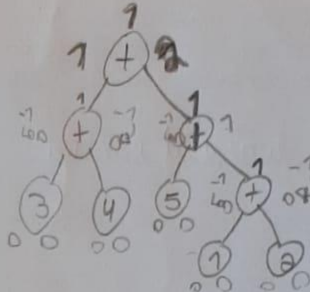
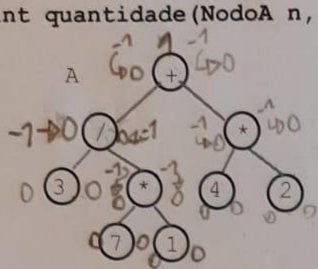
c) Apresente o percurso pós-fixado à esquerda da ABP desenhada na questão 1a.

QUESTÃO 2) (2,5 PONTOS) Considere uma árvore binária que armazena expressões aritméticas, sendo operadores e operandos armazenados como char (se for operando, utiliza-se o seu valor numérico).

Escreva um método/função (utilizando linguagem de programação C, Java) que receba por parâmetro um char que contém um caracter representando um operador (por exemplo '+') e retorne a **quantidade de vezes que este operador aparece na árvore** (por exemplo, na árvore A o operador '+' aparece 1 vez).

Se o operando não estiver na árvore, o método/função deve retornar -1.

int quantidade(NodoA n, char op)



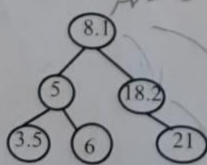
QUESTÃO 3) (2,5 PONTOS) Considere uma ABP arv, que armazena números fracionários representando a temperatura de uma máquina. A ABP já foi criada e já tem valores inseridos (menores na estão à esquerda, maiores à direita).

Implemente (utilize uma linguagem de programação – C, Java) uma função/método que retorne a **média** dos valores armazenados na ABP, **mas descartando o maior valor e o menor valor armazenados**:

float mediatemp(NodoA n)

$$\begin{array}{r} 373 \ 40 \\ - 360 \ 9 \\ \hline 130 \end{array}$$

Exemplo:



Por exemplo, na árvore ao lado, descarta-se 3.5 e 21. A média retornada seria o resultado de $(5+6+8.1+18.2)/4$

num	termos	maior	menor
0	0	8.1	8.1
0	0	8.1	5
5	1	8.1	3.5
11	2	8.1	3.5
12.1	3	18.2	3.5
37.3	4	21	3.5

QUESTÃO 4) (1,5 PONTO) Considere a seguinte expressão pré-fixada que representa uma expressão aritmética armazenada em uma árvore binária:

$/*38*+21,4/$

$38 * (21 + 4) *$

- Desenhe a árvore.
- Escreva essa mesma expressão em notação pós-fixada.
- Qual o resultado da expressão?

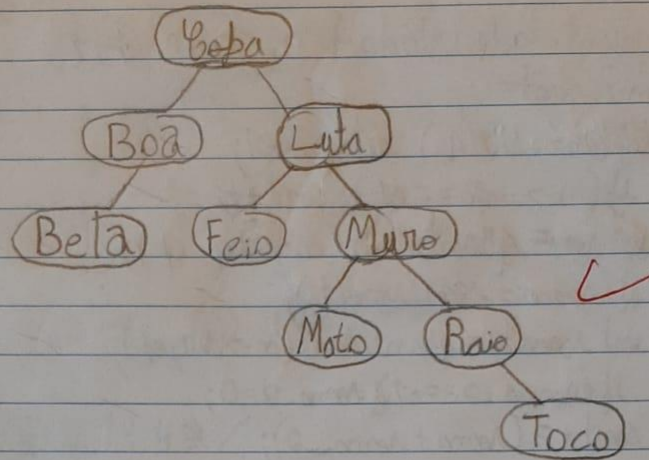
QUESTÃO 5) (1,5 PONTO) Desenhe a árvore AVL resultante da seguinte inserção de nomes:

1,5

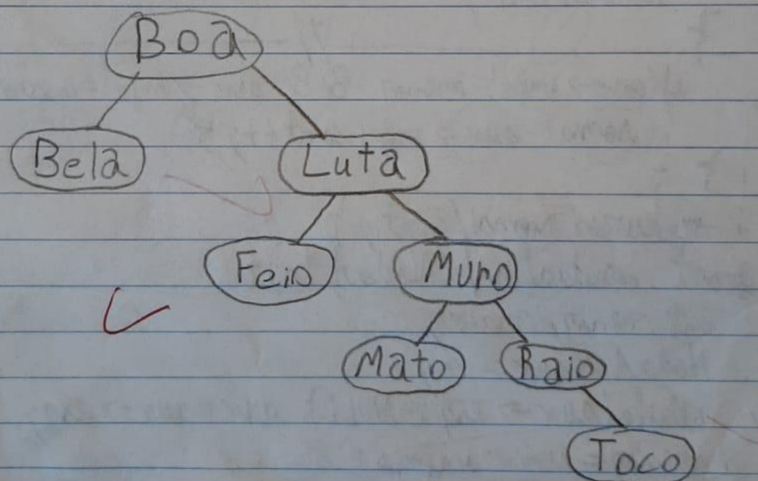
copa – luta – boa – muro – mato – bela – feio – raio - toco

Eduardo Elvhorzlt Pereira

1) a)



b)



c) Bela, Boa, Feio, Mato, Toco, Raio, Muro, Luta, Ceca.

2) int quantidade (NodoA *n, char op) {

int cont=0;

if (n==NULL) return -1;

if (n->info==op) cont++;

int soma = quantidade (n->esq, op);

if (soma == -1) soma = 0;

int soma_2 = quantidade (n->dir, op);

if (soma_2 == -1) soma_2 = 0;

cont += (soma + soma_2);

if (cont == 0) return -1;

return cont;

}

if (aux->info != menor && aux->info != maior) {

mat = aux->info; cont++; }

}

return soma/cont; }

3) float mediatemp (NodoA *n) {

int menor, maior;

NodoA *aux = n;

while (aux->esq != NULL) aux = aux->esq;

menor = aux->info;

aux = n;

while (aux->dir != NULL) aux = aux->dir;

maior = aux->info;

enfileira (n); // inicia fila com a raiz

int soma = 0, cont = 0;

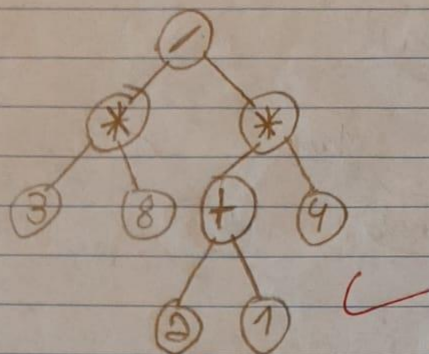
while (fila vazia() != 1) { // enquanto houverem elementos na fila

aux = desenfilar(); // retorna o nó na frente da fila

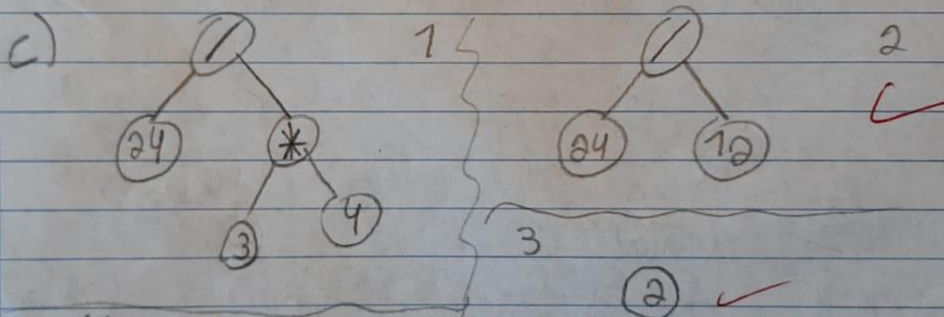
if (aux->esq != NULL) enfileira (aux->esq);

if (aux->dir != NULL) enfileira (aux->dir);

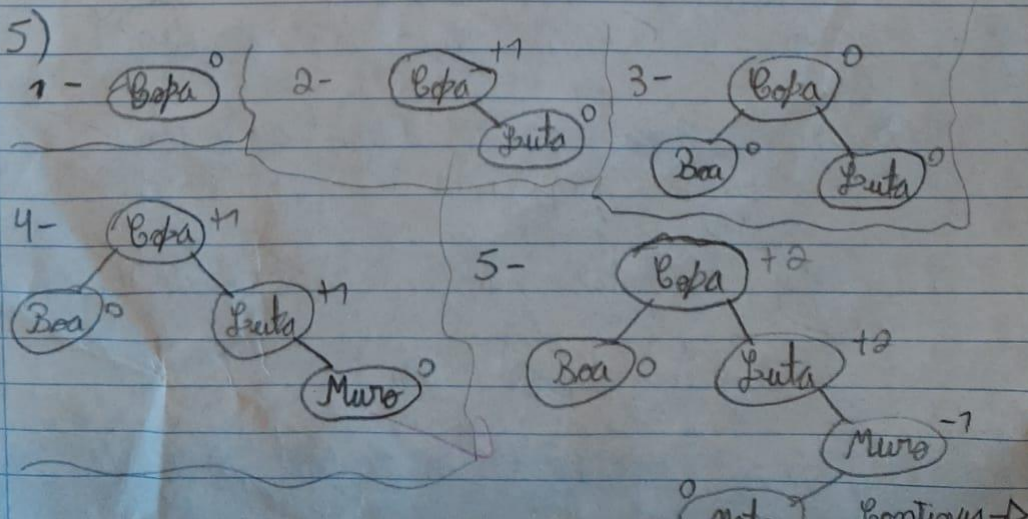
4) a)

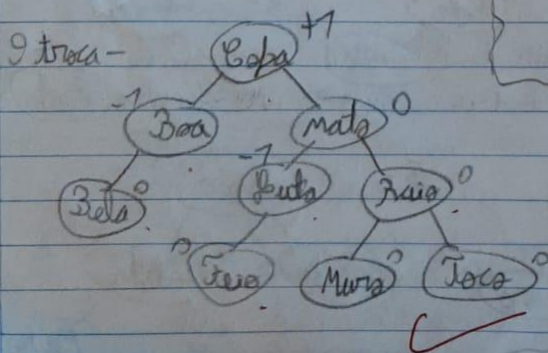
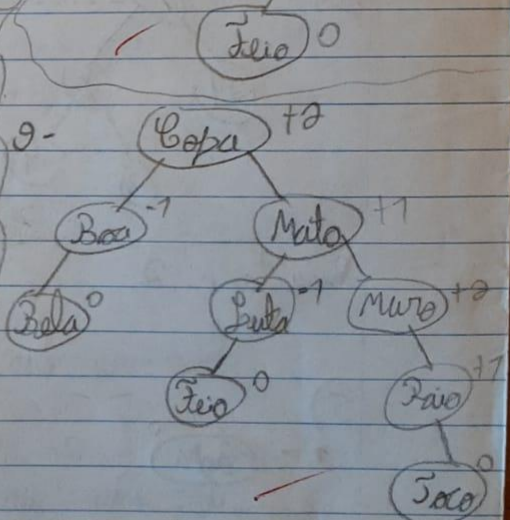
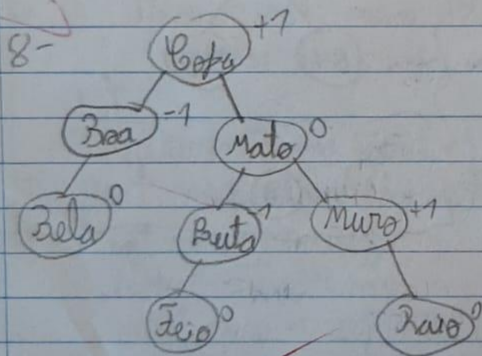
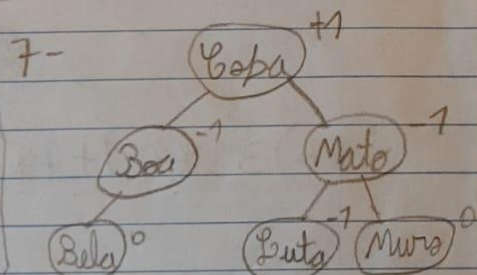
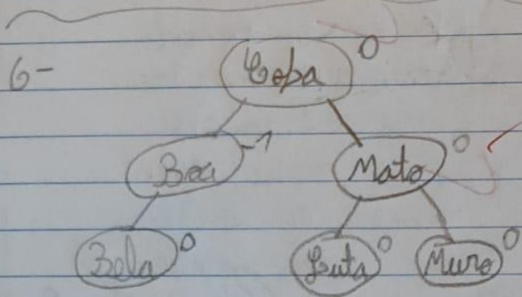
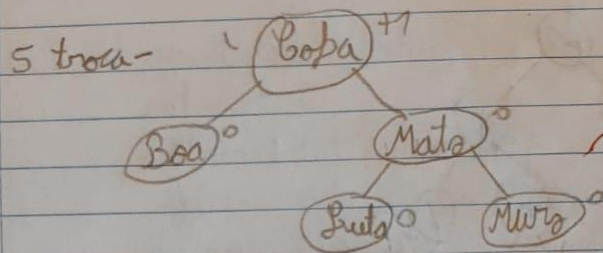


b) $3 * 8 + 2 * 1 + 4 * 1$ ✓



Resultado: 2





← FINAL